Abstracts

Listed are abstracts from recent papers by IBM authors. Inquiries should be directed to the publications cited.

STREMA: Specifying application processes using streams, Ian A. Clark (IBM UK Scientific Centre, Peterlee, England), *The Computer Journal* **21,** No. 1, 25–30 (1978). The search for techniques to shorten application development time and increase the quality of the application has intensified in recent years. The subject of this paper is a method of application design that involves the computer in the actual design procedure. STREMA, a contraction of "stream-schema," is a graphic conversation language that uses streams as basic building blocks. The designer, interacting with the computer, selects streams of input data and procedures to produce a model of the application that changes as the design progresses. The objective of the system is an application that is easy to produce, understand and manipulate. This paper discusses research that uses the concepts of streams within a relational view of data and the use of STREMA in application development.

A development-project reporting and control technique, William Rosenbluth (FSD Gaithersburg, Maryland), *Interface Age* **3,** No. 5, 110–120 (1978). This paper presents a report and control scheme used by the author in controlling development projects. Nine report types are presented and their use in the project review process is described. Illustrations of the reports are included.

Applications of symbolic execution to program testing, James C. King and John A. Darringer (RES Yorktown Heights, NY), *Computer* **11,** No. 4, 51–60 (April, 1978). Program testing techniques can vary from individual test runs to general correctness proofs. Symbolic execution is a program analysis tool which allows the programmer to more effectively prove his program. By replacing real data objects with symbols as input values and executing the program, the symbolic test can be made equivalent to a large number of normal test runs. This paper discusses a symbolic interpretor and how it can be used in program testing situations.

Practical problems in a distributed application, Eric D. Carlson and Mary C. Smyly (RES San Jose, CA), *AFIPS Conference Proceedings, 1978 National Computer Conference* **47,** 823–828, AFIPS Press, Montvale, NJ (1978). This paper discusses the experience obtained from taking an IBM Program Product, Trend Analysis/370, and comparing distributed and nondistributed versions. TA/370 is a distributed application where the color graphics are accomplished on a satellite computer and the data base management and data transformation activity are done on a host computer. An attempt was made to place the graphics package on the host and compare the results. The prototype showed that communications overhead can weaken the theoretical advantage of distributed processing.

Component progress: Its effects on high-speed computer architecture and machine organization, Erich Bloch and Donald J. Galage (DSD, East Fishkill, NY), *Computer* **11,** No. 4, 64–76 (April, 1978). Advances in component technology from vacuum tubes to large-scale integration have had and will continue to have a profound impact on systems architecture and machine design. This progress in component technology has come from developments in semiconductor technology, circuit design and packaging. This paper summarizes these advances and relates them to systems progress. It also discusses future trends in components and the resulting machine designs.

Ciphertext/plaintext and ciphertext/key dependence vs number of rounds for the data encryption standard, Carl H. Meyer (SCD Kingston, NY), *AFIPS Conference Proceedings, 1978 National Computer Conference* **47,** 1119–1126, AFIPS Press, Montvale, NJ (1978). This paper analyzes the intersymbol dependence between ciphertext and plaintext and ciphertext and key bits. A method is shown for determining how fast this dependence builds up. Some introductory information is provided on cryptography and the Data Encryption Standard (DES).

General technique for communications protocol validation, Colin H. West (RES Zurich, Switzerland), *IBM Journal of Research and Development* **22,** No. 4, 393–404, (July, 1978). A technique for the validation of protocols in communications is described. It can be used for systems composed of processes that can be modeled as finite directed graphs. The validation exhaustively exercises the interaction domain of a system and identifies all occurrences of a number of well-defined error conditions. The method can detect when individual processes have no predefined response to incoming messages, as well as system deadlocks and potential loss of messages due to overflow conditions.

Space shuttle software, Caroline T. Sheridan (FSD Houston, TX), *Datamation* **24,** No. 7, 128–140 (July, 1978). Preservation of data integrity in an environment where redundant computers are executing the same real-time control code asynchronously presents significant software challenges. Software for the space shuttle mission had the characteristic of producing identical outputs from several computers running with asynchronous interrupts and priority levels. In addition, the system required a high degree of reliability. This paper discusses the system hardware, the problems faced by software developers, the techniques applied to synchronize the computers and the testing procedures used to prove the system.